AMENDMENTS TO THE CLAIMS:

This listing of the claims will replace all prior versions, and listings, of the claims in this application.

Listing of Claims:

1. (Original) A digital data processor comprising an instruction unit, said instruction unit comprising a code page that is partitioned for storing in a first section thereof a plurality of instruction words and, in association with at least one instruction word, in a second section thereof an extension to said at least one instruction word.

2. (Currently Amended) A digital data processor as in claim 1, where said first section is comprised of a first plurality of contiguous storage locations, and where said second section is comprised of a second plurality of contiguous storage locations, and where said ~~program instructions~~ at least one instruction word and said extension to to said at least one instruction word are one of fixed length and variable length program instructions.

3. (Original) A digital data processor as in claim 1, further comprising at least one page table entry bit having a state for indicating, on a code page by code page basis, whether the code page is partitioned into said first and second sections for storing instruction words and at least one instruction word extension, or whether the code page is comprised instead of a single section storing only instruction words.

4. (Currently Amended) A digital data processor as in claim 3, where said at least one page table entry bit is output from a translation lookaside buffer (TLB).

5. (Original) A digital data processor as in claim 3, further comprising address fault circuitry for determining, in accordance with a state of the at least one page table entry bit, whether a generated instruction address is a valid address for the code page.

6. (Original) A digital data processor as in claim 3, further comprising address circuitry for addressing an instruction word in said first section using a current instruction address, while simultaneously addressing an extension to said instruction word at a fixed offset from said current instruction address.

7. (Original) A digital data processor as in claim 2, where at least some of the second storage locations are not allocated for storing instruction word extensions.

8. (Original) A digital data processor as in claim 7, where said at least some of the second storage locations that are not allocated for storing instruction word extensions are allocated instead for storing at least one of constant values, security information, and error detection and/or correction information for the code page.

9. (Original) A digital data processor as in claim 1, further comprising an address comparator for detecting when program execution has reached the end of the first section for ensuring that a next instruction address is not contained in the second section.

10. (Original) A digital data processor as in claim 1, where each instruction word has a width of x bits, where each extension has a width of y bits, where x=n(8-bits), where y=m(8-bits), where n is an integer greater than one, and where m has a value less than one, equal to one, or greater than one.

11. (Original) A digital data processor as in claim 1, further comprising circuitry, coupled to an output of said code page, to combine an addressed instruction word read out of said code page with a corresponding instruction word extension that is also read out of said code page.

12. (Original) A digital data processor as in claim 11, where said combining circuitry comprises an instruction cache having a bit width w at least equal to a width of an instruction word plus a width of the instruction word extension.

13. (Original) A digital data processor as in claim 11, where said combining circuitry comprises an instruction cache having a bit width w at least equal to a width of an instruction word plus a width of the instruction word extension, said instruction cache having an output coupled to an input stage of an instruction pipeline, said input stage having a bit width of w.

14. (Original) A digital data processor as in claim 11, where said combining circuitry comprises an input stage of an instruction pipeline.

15. (Original) A digital data processor as in claim 11, where said combining circuitry

comprises an instruction decode stage of an instruction pipeline.

16. (Original) A digital data processor as in claim 3, further comprising circuitry, coupled to an output of said code page, to selectively combine, in response to the state of said at least one page table entry bit, an addressed instruction word read out of said code page with a corresponding instruction word extension that is also read out of said code page.

17. (Original) A digital data processor as in claim 16, where said combining circuitry comprises a multiplexer having a first set of inputs coupled to an instruction word extension output of said code page and a second set of inputs coupled to an invalid instruction word extension.

18. (Original) A digital data processor as in claim 16, where said combining circuitry comprises an instruction decode stage of an instruction pipeline.

19. (Currently Amended) A digital data processor as in claim 1, where said instructions words comprise Reduced Instruction Set Computer (RISC) instructions.

20. (Currently Amended) A digital data processor as in claim 19, where said ~~RISC~~ instructions words have a width of 32-bits, where said ~~at least one~~ extension to said at least one instruction word has a width of 8-bits.

21. (Original) A digital data processor as in claim 20, where said code page has a storage capacity of 4096 bytes, where said first section comprises 3072 bytes, and where said second section comprises 1024 bytes.

22. (Original) A method to operate an instruction unit having a code page, comprising:

partitioning said code page into at least two sections; and

storing in a first section thereof a plurality of instruction words and, in association with at least one instruction word, storing in a second section thereof an extension to said at least one instruction word.

23. (Original) A method as in claim 22, where said first section is comprised of a first plurality of contiguous storage locations, and where said second section is comprised of a second plurality of contiguous storage locations.

24. (Original) A method as in claim 22, further comprising setting a state of at least one page table entry bit for indicating, on a code page by code page basis, whether the code page is partitioned into said first and second sections for storing instruction words and at least one instruction word extension, or whether the code page is comprised instead of a single section storing only instruction words.

25. (Currently Amended) A method as in claim 24, further comprising outputting said at least one page table entry bit from a translation lookaside buffer (TLB).

26. (Original) A method as in claim 24, further comprising determining, in accordance with a state of the at least one page table entry bit, whether a generated instruction address is a valid address for the code page.

27. (Original) A method as in claim 24, further comprising addressing an instruction word in said first section using a current instruction address, while simultaneously addressing an extension to said instruction word at a fixed offset from said current instruction address.

28. (Original) A method as in claim 23, where at least some of the second storage locations are not allocated for storing instruction word extensions.

29. (Original) A method as in claim 28, where said at least some of the second storage locations that are not allocated for storing instruction word extensions are allocated instead for storing at least one of constant values, security information, and error detection and/or correction information for the code page.

30. (Original) A method as in claim 22, further comprising detecting when program execution has reached the end of the first section for ensuring that a next instruction address is not contained in the second section.

31. (Original) A method as in claim 22, where each instruction word has a width of x bits, where each extension has a width of y bits, where x=n(8-bits), where y=m(8-bits), where n is an integer greater than one, and where m has a value less than one, equal to one, or greater than one.

32. (Original) A method as in claim 22, further comprising combining an addressed instruction word read out of said code page with a corresponding instruction word extension that is also read out of said code page.

33. (Original) A method as in claim 32, where combining operates an instruction cache having a bit width w at least equal to a width of an instruction word plus a width of the instruction word extension.

34. (Original) A method as in claim 32, where combining operates an instruction cache having a bit width w at least equal to a width of an instruction word plus a width of the instruction word extension, said instruction cache having an output coupled to an input stage of an instruction pipeline, said input stage having a bit width of w.

35. (Original) A method as in claim 32, where combining occurs at an input stage of an instruction pipeline.

36. (Original) A method as in claim 32, where combining occurs at an instruction decode stage of an instruction pipeline.

37. (Original) A method as in claim 24, further comprising selectively combining, in response to the state of said at least one page table entry bit, an addressed instruction word read out of said code page with a corresponding instruction word extension that is also read out of said code page.

38. (Original) A method as in claim 37, where selectively combining comprises operating a multiplexer having a first set of inputs coupled to an instruction word extension output of said code page and a second set of inputs coupled to an invalid instruction word extension.

39. (Original) A method as in claim 37, where selectively combining comprises operating an instruction decode stage of an instruction pipeline.

40. (Currently Amended) A method as in claim 22, where said instructions words comprise Reduced Instruction Set Computer (RISC) instructions.

41. (Currently Amended) A method as in claim 40, where said ~~RISC~~ instructions words have a width of 32-bits, where said ~~at least one~~ extension to said at least one instruction word has a width of 8-bits.

42. (Original) A method as in claim 41, where said code page has a storage capacity of 4096 bytes, where said first section comprises 3072 bytes, and where said second section comprises 1024 bytes.

43. (Original) A computer program stored on a computer readable medium, said computer program comprising instructions for use with an instruction unit having a code page, comprising:

computer program code for partitioning said code page into at least two sections for storing in a first section thereof a plurality of instruction words and, in association with at least one instruction word, for storing in a second section thereof an extension to said at least one instruction word; and

computer program code for setting a state of at least one page table entry bit for indicating, on a code page by code page basis, whether the code page is partitioned into said first and second sections for storing instruction words and at least one instruction word extension, or whether the code page is comprised instead of a single section storing only instruction words.

44. (Original) A computer program as in claim 43, further comprising computer program code for ensuring that a last instruction in said first section is a branch instruction the execution of which does not specify a target address that lies in the second section.